6 Remote memory access protocol (normative)

6.1 General

NOTE FOR REVIEWERS: The indirect read and write commands have been removed, the validate before write option has been added, and a read-modify-write command has been included.

The padding of data fields to 32-bits is to be considered. For example, we could allow padding of the data field to the next 32-bit boundary with the data length giving the correct number of bytes to be transferred. E.g. if three bytes are to be sent then an additional zero byte could be added and data length set to 3. Up to three pad bytes could be added without causing a data length error.

The possibility of sending a reply if there is an error but not if there is no error is also to be considered. At the moment the Ack/No_Ack bit has two possibilities: Ack and error indication or no ack and no error indication. The third possibility is to not provide acknowledgements but to provide an error indication.

The remote memory access protocol (RMAP) is used to write to and read from memory or registers in a destination node on a SpaceWire network. Input/output registers and control/status registers are assumed to be memory mapped so are accessed as memory.

All read and write operations defined in the RMAP protocol are posted operations i.e. the source does not wait for an acknowledgement or reply to be received. This means that many reads and writes can be outstanding at any time. It also means that there is no timeout mechanism implemented in RMAP for missing acknowledgements or replies. If an acknowledgement or reply timeout mechanism is required it must be implemented in the source user application.

6.1.1 Write commands

Writes commands can be acknowledged or not acknowledged by the destination node when they have been received correctly. If the write is to be acknowledged and there is an error with the write request, the destination will send an error code to the source that sent the command. The error can only be sent to the source if the write command header was received intact, so that the destination that detected the error knows where to send the error message.

Write commands can perform the write operation after verifying that the data has been transferred to the destination without error, or it can write the data without verification. To perform verification on the data requires buffering in the destination node to store the data while it is being verified, before it is written. The amount of buffering is likely to be limited so verified writes ought only be performed for relatively short sets of data, that will fit in the available buffer at the destination. Longer writes can be performed but without verification prior to writing. Verification in this case is done after the data has been written. Verified writes should always be used when writing to configuration or control registers.

The acknowledged/non-acknowledged and verified/non-verified options to the write command result in four different write operations:

- **Write non-acknowledged, non-verified** – writes zero or more bytes to memory in a destination node. The command is checked using a checksum before the data is written, but the data itself is not checked before it is written. No acknowledgement to indicate that the command has been executed is sent to the source of the write command. This command is typically used for writing large amounts of data to a destination where it can be safely assumed that the write operation completed successfully.
- **Write non-acknowledged, verified** – writes zero or more bytes to memory in a destination node. Both the command and data are checked using checksums before the data is written. This limits the amount of data that can be transferred in a single write operation, but erroneous data cannot be written to memory. No acknowledgement to indicate that the command has been executed is sent to the source of the write command. This command is typically used for writing command registers and small amounts of data to a destination where it can be safely assumed that the write operation completed successfully.
- **Write acknowledged, non-verified** – writes zero or more bytes to memory in a destination node. The command is checked using a checksum before the data is written, but the data itself is not checked before it is written. An acknowledgement to indicate that the command has been executed is sent to the source of the write command. This command is typically used for writing large amounts of data to a destination where it can be safely assumed that the write operation completed successfully, but an acknowledgement is required. For example writing sensor data to memory.
- **Write acknowledged, verified** – writes zero or more bytes to memory in a destination node. Both the command and data are checked using checksums before the data is written. This limits the amount of data that can be transferred in a single write operation, but erroneous data cannot be written to memory. An acknowledgement to indicate that the command has been executed is sent to the source of the write command. This command is typically used for writing small amounts of data to a destination where it is important to have confirmation that the write operation was executed successfully. For example writing to command or configuration registers.

6.1.2 Read commands

The read command reads one or more bytes of data from a specified area of memory in a destination node. The data read is returned in a reply packet.

6.1.3 Read-modify-write

The read-modify-write command reads a register (or memory) returning its value and then writes a new value, specified in the command, to the register. A mask can be included, in the command, so that only certain bits of the register are written. This provides an atomic operation that can be used for semaphores and other handshaking operations.

6.1.4 Guide to clause 6

A set of definitions is given in sub-clause 6.2. The various write commands are defined in sub-clause 6.3. The read command is described in sub-clause 6.4, and the read-modify-write command in sub-clause 6.5.

6.2 Definitions

Path Address is a SpaceWire path address which defines the route to a destination node by specifying, for each router encountered on the way to the destination, the output port that a packet is to be forwarded through. A path address comprises one byte for each router on the path to the destination. Once a path address byte has been used to specify an output port of a router it is deleted to expose the next path address byte for the next router. All path address bytes will have all been deleted by the time the packet reaches the destination

Logical Address byte is the logical address of the destination. This may be used to route the packet to the destination or, if path addressing is being used, to simply confirm that the final destination is the correct one i.e. that the logical address of the destination matches the logical address in the packet. If the logical address of the destination is unknown then the default logical address of 254 may be used (see sub-clause 5.2.1). The destination may chose to accept or reject packets with a logical address of 254.

Protocol Identifier byte identifies the particular protocol being used for communication. For the Remote Memory Access protocol the protocol identifier has the value 1 (01h).

Packet Type, Command, Source Address Length byte determines the type of the packet i.e. a command, a response or an acknowledgement. This byte also includes two bits that determine the number of extra 4-byte return addresses. For example, if these bits are set to the value two then there will be eight extra source address bytes. If they are set to zero then there are no extra address bytes.

Device Type defines the type of device that is expected to be the destination of a command. For a reply to a command it indicates the type of device that will send the reply. This provides a level of security to command execution. The Device Type byte must match the type of device receiving a command or the command will not be executed.

Extra Source Address bytes provide extra bytes for the reply or acknowledgement to a command. The source address is used by the destination node to send acknowledgements or data read back to the source that requested a write or read operation. The Extra Source Address byte allows path addressing and regional logical addressing to be used to specify the source node. Leading zeros of the return address are ignored. If a packet is to be sent to address zero then this is done by setting all the extra return address bytes to zero. This will result in a single zero address byte being sent in front of the source address.

Source Address byte is the logical address to which the destination node for a command is to reply. The Source Address is normally set to the logical address of the source node that is sending the command. The Source Address byte may be set to 254 (0FEh) which is the default logical address, if the command source node does not have a logical address.

Transaction Identifier bytes are used to identify command, response, and acknowledge transactions that make up a particular read or write operation. The source of the command gives the command a unique transaction identify. This transaction identifier is returned in the response or acknowledgement to the command. This allows the command source to send many commands without having to wait for a response to each command before sending the next command. When a response or acknowledge comes in it can be quickly associate with the command that caused it by the transaction identifier.

Extended Address byte is used to extend the 32-bit memory address to 40-bits allowing a 1 Terabyte address space to be accessed directly in each node. For nodes that do not support a 40-bit address space this byte should be set to zero.

Memory Address bytes form the bottom 32-bits of the memory address to which the data in a write command is to be written or from where data is to be read for a read command. Input/output registers and control/status registers are assumed to be memory mapped.

Data Length bytes form the 24-bit length of the data that is to be written or read. The length is the length in bytes with the most-significant byte of the length sent first.

Header Checksum byte is an 8-bit checksum used to confirm that the header is correct before executing the command. The header checksum is formed using modulo 256, unsigned addition of the bytes starting with the destination logical address and ending with the header checksum itself. The header checksum is set so that the addition total is zero.

Data bytes are the data that is to be written in a write command or the data that is read in a read response.

Data Checksum byte is an 8-bit checksum used to confirm that the data is correct before being written in a verified write command or was correctly transferred in a non-verified write command or read reply. The data checksum starts with the byte after the header checksum and is calculated in the same way as the header checksum so that the sum of all the data bytes and the data checksum is zero.

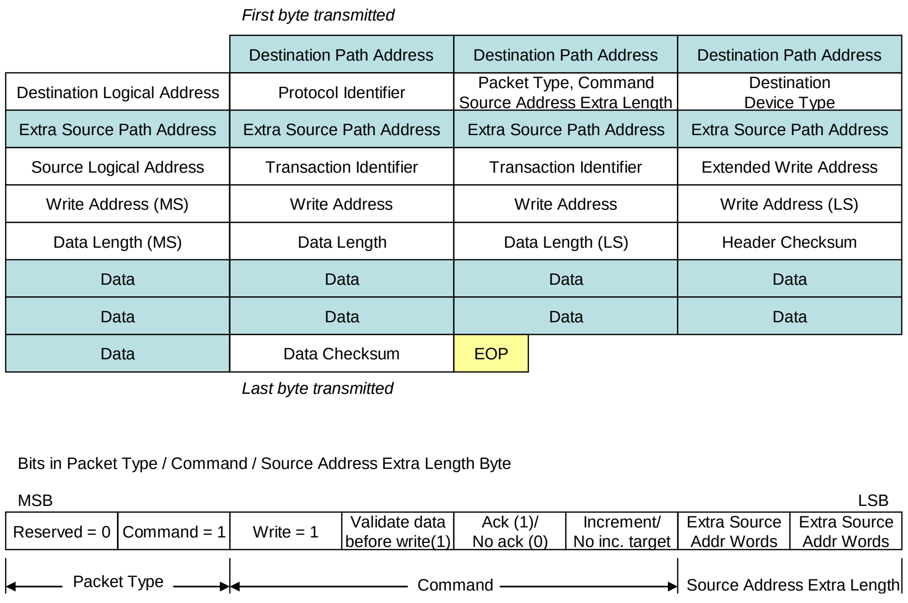
EOP character is the End Of Packet market of the SpaceWire packet.

6.3 Write Command

The various types of write command are describe here.

6.3.1 Write command format

The write command provides a means for one node, the source node, to write one or more bytes of data into memory of another node, the destination node on a SpaceWire network. The format of the command is shown in Figure 6-1.



First byte transmitted

	Destination Path Address	Destination Path Address	Destination Path Address
Destination Logical Address	Protocol Identifier	Packet Type, Command Source Address Extra Length	Destination Device Type
Extra Source Path Address	Extra Source Path Address	Extra Source Path Address	Extra Source Path Address
Source Logical Address	Transaction Identifier	Transaction Identifier	Extended Write Address
Write Address (MS)	Write Address	Write Address	Write Address (LS)
Data Length (MS)	Data Length	Data Length (LS)	Header Checksum
Data	Data	Data	Data
Data	Data	Data	Data
Data	Data Checksum	EOP	

Last byte transmitted

Bits in Packet Type / Command / Source Address Extra Length Byte

MSB							LSB
Reserved = 0	Command = 1	Write = 1	Validate data before write(1)	Ack (1)/ No ack (0)	Increment/ No inc. target	Extra Source Addr Words	Extra Source Addr Words
Packet Type		Command				Source Address Extra Length	

Figure 6-1 Write Command Format

The Destination Path Address is the address on the SpaceWire network of the node that is to have data written into its memory. The destination address is made up of two parts: the Destination Path Address bytes which are optional (shaded in Figure 6-1) and the Logical Address. If path addressing is being used then the Destination Address bytes contain the path to the destination node. The Destination Logical Address byte is then set to the logical address of the destination node or to the default value 254 (0FEh). If logical addressing is being used there are no Destination Address bytes and the Destination Logical Address is set to the logical address of the destination node. Normally logical addressing would be used and there would be no Destination Address bytes.

The Protocol Identifier byte is set to the value 1 (01h) which is the Protocol Identifier for the Remote Memory Access protocol.

The Packet Type field comprises a reserved bit and a command/response bit which is set (1) for a command and clear (0) for a response. The packet type field for the write command is 01b, i.e. the command/response bit is set, to indicate that the packet is a command packet, rather than a reply packet. The reserved bit is clear (0).

The Command field holds the direct write command.

The Read/Write bit is zero for a write command.

The Validate Data Before Write bit is set (1) if the data is to be validated before it is written to memory. The command header is always checked using a checksum (Header Checksum see below) before the command is executed. If the Validate Data Before Write bit is set then the entire command must be buffered and validated using the Header Checksum and the Data Checksum before the command is executed. Since the entire command and data has to be buffered this places a limit on the amount of data that can be included in the write command. All RMAP compliant interfaces have to support the buffering and validation of write command with at least four bytes of data. The buffering and validation of write commands with more than four bytes of data is dependent on the particular interface. If the Validate Data Before Write bit is not set (0) then the data is not validated before it is written. This enables much larger amounts of data than can be buffered to be written in a single command. The command header is validated with the Header Checksum so that it is confirmed that the correct memory address and data length is being used. The data is then streamed into the memory space as it arrives without first being checked. Once all the data has been written to the specified memory area the data is validated using the Data Checksum. This is acceptable because even if the wrong data has been written to memory, at least it has not been written in the wrong place. The error will be reported to the source node if the Ack/No_Ack bit has been set (1) to request an acknowledgement to the write command. If the source is able to resend the data then this can be done. When writing to control and configuration registers it is essential that the Validate Data Before Write bit is set (1).

The Ack/No_Ack bit is set (1) if an acknowledgement to the write command is required and cleared (0) if no acknowledgement is to be sent. If no acknowledgement is requested then the source will not be informed when an error occurs in the write command.

The command option “Increment / No Increment Target” is used for multiple data byte transfers. If set (1) it causes the write memory address in the target to increment on every byte (or word as determined by the target unit) written so that data bytes are written to consecutive memory locations. If not set (0) the write memory address is not incremented so successive data bytes (or words as determined by the target unit) are written to the same memory location. Note that the width of the memory word is determined by the target unit and can be any multiple of 8-bits. For example, if the width of the target unit memory word is 32-bits then four data bytes from the data field of the command are written into one memory location in the target unit.

The Source Address Length field is set to zero if logical addressing is being used. If path addressing (or regional logical addressing) is being used then the Source Address Length field has to be set to the smallest number of 32-bit words that can be used to contain the path address from the destination node that is being written to back to the source of the command packet. For example, if three path address bytes are required then the Return Address Length field is set to one.

The Device Type byte contains an eight-bit code representing the type of SpaceWire device. For example, if the Device Type is set to 01h then the device that is being written to should be a SpaceWire router configuration port. The Device Type field provides a level of security. Commands with a Device Type that does not match the type of the destination node will not be executed.

The Extra Source Path Address bytes contain any required path address (or regional logical address) bytes needed to route the reply packet from the destination node back to the source node. If logical addressing is being used then the Extra Source Address bytes are not present.

The Source Logical Address byte contains the logical address of the source of the write command packet. If the source node does not have a logical address because only path addressing is being used then the Source Logical Address byte must be set to 254 (0FEh) (see sub-clause 5.2.1) which is the default logical address.

The Transaction Identifier bytes are set to the value provided by the user application in the source node. Typically transaction identifiers are an incrementing integer sequence, with each successive RMAP transaction being given the next number in the sequence. The intention of the transaction identifier is to uniquely identify a transaction. The reply to a write command contains the same transaction identifier as in the write command. Thus it can be readily matched, by the user application in the source node, to the specific command that caused the reply.

The Extended Memory Address byte holds the most-significant 8-bits of the memory address to be written to. This extends the 32-bit memory address to 40-bits allowing access to 1 Terabyte of memory space in each node. This byte is set to zero if extended memory addressing is not being used.

The four Memory Address bytes hold the bottom 32-bits of the memory address to which the data in a write command is to be written. The first byte sent in the command is the most significant byte of the address. When combined with the Extended Memory Address byte a 40-bit memory address is provided.

The three Data Length bytes contain the length of the data that is to be written. This gives a maximum data length of 16 Mbytes in a single write command. If a single byte is being written this field is set to one. If set to zero then no bytes will be written to memory which may be used as a test transaction. The first byte sent is the most significant byte of the data length.

The Header Checksum byte is an 8-bit checksum used to confirm that the header is correct before executing the command.

The Data bytes contain the data that is to be written into the memory of the destination node. When writing to memory organised in words (e.g. 32-bit words) then the first byte sent is the most-significant byte of the word.

The Data Checksum byte contains an 8-bit checksum used to confirm that the data was correctly transferred. In a write command data is written to target memory provided that the header checksum shows no error in the header. This helps to prevent inadvertent writing to incorrect areas of memory when there is an error in the header. If there is an error in the data checksum then the wrong data will have been written to memory, but is will not have been written to the wrong place. The user application at both source and destination will be informed that there was an error in the data transferred so that corrective action can be taken.

EOP character is the End Of Packet market of the SpaceWire packet.

6.3.2 Write reply format

The reply to a write command is sent by the destination back to source of the write command. The reply is used to indicate the success or failure of the write command. The format of the write reply is shown in Figure 6-2.

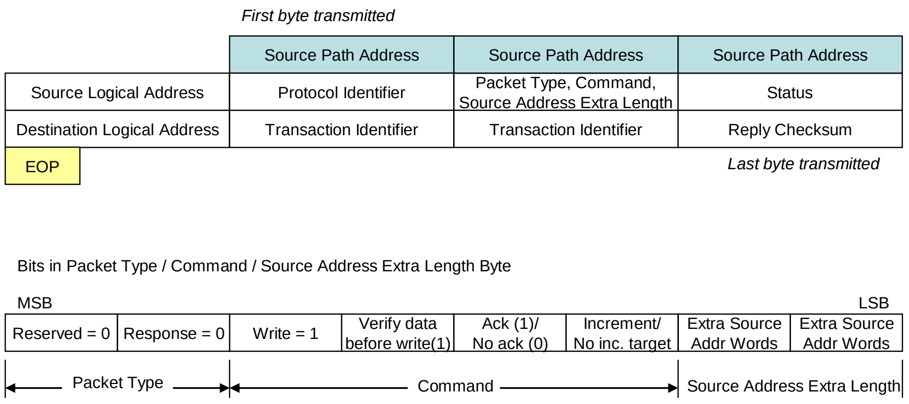


Figure 6-2 Write Reply Format

The Source Path Address bytes contain any required path address bytes needed to route the reply packet from the destination node back to the source node. The value of the Source Path Address bytes are as specified in the Extra Source Path Address field of the write command. If logical addressing is being used then the Source Path Address bytes are not present in the reply to the write command. Any Source Path Address bytes are stripped off by the time the reply reaches the source of the write command.

The Source Logical Return Address byte contains the logical address of the source of the write command packet, as specified in the write command Source Address field.

The Protocol Identifier byte is set to the value 1 (01h) which is the Protocol Identifier for the Remote Memory Access protocol.

The Packet Type field is 00b to indicate that this is a reply packet.

The Command and Source Address Length field are set to the same values as in the command byte of the write command.

The Status byte provides the status of the write command. This is set to zero if the command executed successfully and to a non zero error code if there was an error. See error codes sub-clause 6.6.

The Transaction Identifier bytes are set to the same value as provided in the write command. This is so that the source of the write command can associate the reply with the original write command.

The Reply Checksum byte is an 8-bit checksum used to confirm that the reply packet has been received without error. This is calculated in the same way as a header checksum.

6.3.3 Write action

The operation of the write command is illustrated in the sequence diagram of Figure 6-3.

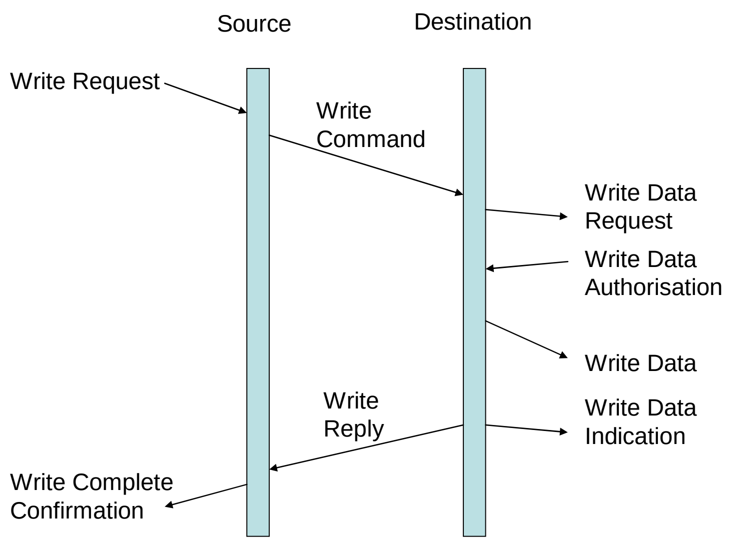


Figure 6-3 Write Command/Acknowledge Sequence

The write command sequence begins when an application requests to perform a write operation (Write Request). In response to this the source node builds the write command and sends it across the SpaceWire network to the destination node (Write Command). When the Write Command arrives at the destination, the header is first checked for errors and if there are no errors the user application at the destination node is asked if it will accept the write operation (Write Data Request). Assuming that authorisation is given by the destination user application (Write Data Authorisation) the data contained in the write command is written into the specified memory location of the destination node (Write Data). If the Validate Data Before Write bit is set in the command field of the header then the data is buffered and checked using the data checksum before it is written to memory.

Once data has been written to memory the user application running on the destination node is informed that a write operation has taken place (Write Data Indication). If an acknowledgement has been requested by setting the Ack/No_Ack bit in the command field then the destination node will wait until the data has been written to memory in the destination node. It will then send a write reply packet back to the source of the write command (Write Reply). When the write reply is received, the source node indicates successful completion of the write request (Write Complete Confirmation).

If no acknowledgement is requested then the destination node waits for the data to be written into destination memory, but does not send an acknowledgement write reply to the source.

Note that the speed with which the destination user application responds to the Write Data Request with a Write Data Authorisation will limit the rate at which RMAP commands can be processed by the destination node. The SpaceWire interface will block during this period. In some cases, for example writing to control or configuration registers, the Write Data Request and Write Data Indication are implicit in the actual write operation so there is no appreciable delay and one command can immediately follow the previous one.

6.3.4 Write errors

There are four principal types of error that can arise during a write operation: Write Command Header Error, Write Authorisation Rejection, Write Command Data Error and Write Reply Error.

The sequence of events that occurs when there is an error in the header of the write command is illustrated in Figure 6-4.

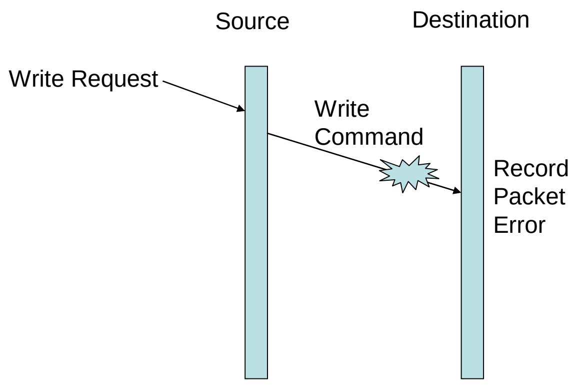


Figure 6-4 Write Command Header Error

The Write Command packet arrives at the destination and its header is found to be in error. This fact is added to the error statistics in the destination node. The remainder of the packet is discarded. No other action is taken at the destination node, specifically no data is written into the memory of the destination node and no write reply packet is sent back to the source node. The source node does not receive a write reply packet so no action is taken by the RMAP protocol in the source node. The user application on the source node may set a timeout time when it requests RMAP to send the write command. When no reply is received this timer will time out and detect the fact that no write reply has been received in the time expected. It is up to the user application in the source node to provide any command reply timeout timers. This is not part of RMAP's responsibilities.

The reason for this is that if RMAP is made responsible for the timeout timers and if posted commands are to be implemented (i.e. many outstanding write commands) then separate timeout timer and reply-received flags will be required for each outstanding write request. This could be a large number and is very much application dependent. Hence the decision to put this responsibility in the user application at the source node. This user application knows how many outstanding requests it will need and can provide both posted and non-posted write operations.

If the write command header is valid, the user application at the destination node is asked if it will accept the write operation. If it rejects the write operation then a write error reply is returned to the source node (assuming that the Ack/No_Ack bit is set in the write command, requesting an acknowledgement or error code to be sent). This situation is illustrated in Figure 6-5. When the Write Reply containing the error code is received back at the source node, a write error indication (Write Data Failure) is signalled to the user application in the source node.

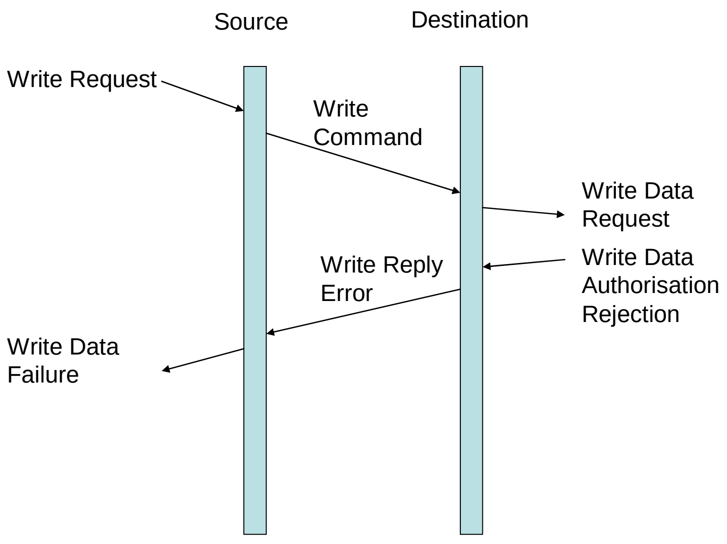


Figure 6-5 Write Data Authorisation Rejection

The situation that arises when there is an error in the data field of the write command is shown in Figure 6-6.

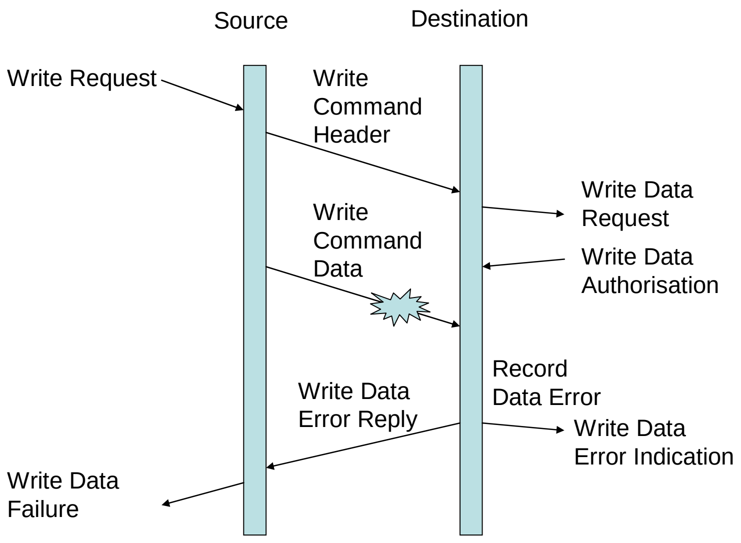


Figure 6-6 Write Command Data Error

Since the header of the write command has been received without error, a request is made to write data to destination node memory (Write Data Request). This request is granted (Write Data Authorisation) and RMAP starts to transfer data from the data field of the received packet into destination node memory. If there is insufficient data in the data field (i.e. the data field is shorter than the data length provided in the write command header) then when the EOP is reached data will stop being transferred into destination memory and an error flag will be raised. Note that in this case the data checksum will also be transferred to memory. If there is too much data in the data field then the specified amount of data, defined by the data length field of the write command header, will be transferred to memory, the rest of the packet will be discarded and an error flag will be raised. If there is a data checksum error then an error flag will be raised after the data has been transferred to destination memory. These various errors will be reported to the user application running on the destination node (Write Error Indication).

Since the header of the write command was intact it is possible to report the error back to the source. A write reply packet is sent back to the source node indicating the type of error that has occurred (Write Data Error Reply). When this is received at the source node the error is reported to the application that requested the write command (Write Data Failure).

It is possible that the write reply is corrupted or for some other reason does not reach the source node intact. This situation is illustrated in Figure 6-7.

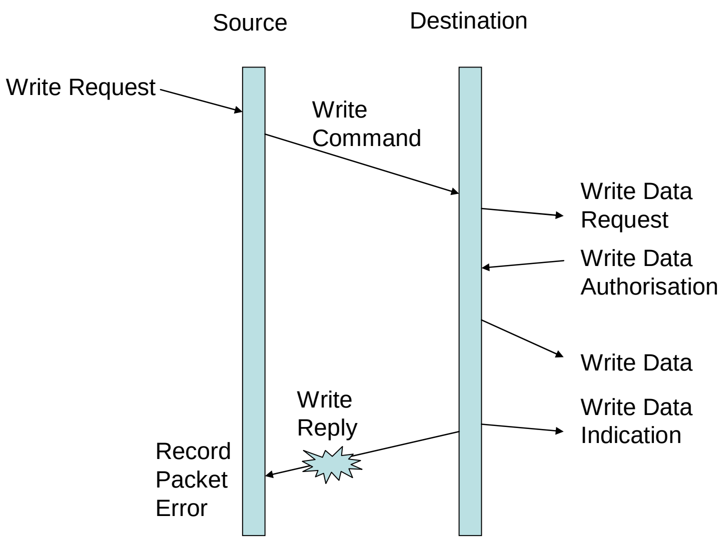


Figure 6-7 Write Reply Error

The data has been correctly written into destination memory and the destination application has been informed. The write reply that is sent back to the source node is corrupted. If the corrupted packet arrives at the source node (or indeed any other node) it is recorded as a packet receive error.

There are no timeout timers for the write requests within the RMAP protocol. Any timing of the write reply must be done by the user application. This is because if timeout timers are contained within RMAP there would have to be one timer per outstanding write operation, which could be a large number of timers in some cases. Just one timer would be possible but the write operation would then be “non-posted” so that there could only be one outstanding write request at any one time. The need to support many outstanding write requests is important as is the need to minimise the hardware needed to implement RMAP, hence the decision to make any write reply timeout the responsibility of the application in the source node.

RMAP informs the application when a write acknowledge is received. It is not responsible for informing the user application if no acknowledge is received.

6.3.5 Write request parameters

The Write Request has to provide the following parameters:

- Destination address
- Source address
- Transaction identifier
- Destination device type
- Write command options
- Write address

- Data length
- Data

6.4 Read Command

6.4.1 Read command format

The read command provides a means for one node, the source node, to read one or more bytes of data from the memory of a destination node. The format of the command is shown in Figure 6-8.

First byte transmitted

	Destination Path Address	Destination Path Address	Destination Path Address
Destination Logical Address	Protocol Identifier	Packet Type, Command Source Address Extra Length	Destination Device Type
Extra Source Path Address	Extra Source Path Address	Extra Source Path Address	Extra Source Path Address
Source Logical Address	Transaction Identifier (MS)	Transaction Identifier (LS)	Extended Read Address
Read Address (MS)	Read Address	Read Address	Read Address (LS)
Data Length (MS)	Data Length	Data Length (LS)	Header Checksum
EOP			

Last byte transmitted

Bits in Packet Type / Command / Source Address Extra Length Byte

MSB LSB

Reserved = 0	Command = 1	Read = 0	Read = 0	Read = 1 (Ack/No_Ack)	Increment/ No inc. target	Extra Source Addr Words	Extra Source Addr Words
Packet Type		Command				Source Address Extra Length	

Figure 6-8 Read Command Format

The Destination Address is the address on the SpaceWire network of the node from which data is to be read. The destination address is made up of two parts: the Destination Path Address bytes which are optional (shaded in Figure 6-1) and the Destination Logical Address. If path addressing is being used then the Destination Path Address bytes contain the path to the destination node. The Destination Logical Address is byte is then set to the logical address of the destination node or to the default value 254 (0FEh). If logical addressing is being used there are no Destination Address bytes and the logical address is set to the logical address of the destination node. Normally logical addressing would be used and there would be no Destination Path Address bytes.

The Protocol Identifier byte is set to the value 1 (01h) which is the Protocol Identifier for the Remote Memory Access protocol.

The Packet Type field is set to 01b indicate that the packet is a command packet, rather than a reply packet.

The Command field holds the read command.

Read/Write bit is clear (0) to indicate that it is a read command.

Validate before write is clear (0) as there is no writing of data.

Ack/No_Ack is set (1) to indicate that a reply will be generated which will contain the data read.

The command option "Increment / No Increment Target" is used for multiple data byte transfers. If set (1) it causes the read address in the target to be incremented after every byte (or word as determined by the target unit) has been read so that data bytes are read from consecutive memory locations. If not set (0) the read address is not incremented so successive data bytes (or words as determined by the target unit) are read from the same memory location. Note that the width of the memory word is determined by the target unit and can be any multiple of 8-bits. For example, if the width of the target unit memory word is 32-bits then four data bytes from the data field of the command are read from one memory location in the target unit.

The Source Address Length field is set to zero if logical addressing is being used. If path addressing is being used then the Return Address Length field has to be set to the smallest number of 32-bit words that can be used to contain the path address from the destination node that is being written to back to the source of the command packet. For example, if three path address bytes are required then the Return Address Length field is set to one.

The Device Type byte contains an eight-bit code representing the type of SpaceWire device. For example, if the Device Type is set to 01h then the device that is being read should be a SpaceWire router configuration port. The Device Type field provides a level of security. Commands with a Device Type that does not match the type of the destination node will not be executed.

The Extra Source Address bytes contain any required path address bytes needed to route the reply packet from the destination node back to the source node. If logical addressing is being used then the Extra Source Address bytes are not present.

The Source Logical Address byte contains the logical address of the source of the read command packet. If the source node does not have a logical address because only path addressing is being used then the Source Logical Address byte must be set to 254 (0FEh) which is the default logical address.

The Transaction Identifier bytes are set to the next transaction identifier in the sequence held by the source node. This uniquely identifies the transaction being started by the read command. The reply to the read command will contain the same transaction identifier and can thus be readily matched to the specific command that caused the reply.

The Extended Read Address byte holds the most-significant 8-bits of the memory address to be read. This extends the 32-bit memory address to 40-bits allowing access to 1 Terabyte of memory space in each node. This byte is set to zero if extended memory addressing is not being used.

The four Read Address bytes hold the bottom 32-bits of the memory address from which data is to be read. The first byte sent in the command is the most significant byte of the address.

The three Data Length bytes contain the length, in bytes, of the data that is to be read. If a single byte is to be read this field is set to one. If set to zero then no bytes will be read from memory which may be used as a test transaction. The first byte sent is the most significant byte of the data length.

The Header Checksum byte is an 8-bit checksum used to confirm that the header is correct before executing the command.

EOP character is the End Of Packet market of the SpaceWire packet.

6.4.2 Read reply format

The read reply contains either the data that was read from the destination node, or an error code indicating why data could not be read. The reply to a read command is sent by the destination node back to the source of the read command. The format of the read reply is illustrated in Figure 6-9.

First byte transmitted

	Source Path Address	Source Path Address	Source Path Address
Source Logical Address	Protocol Identifier	Packet Type, Command, Source Address Extra Length	Status
Destination Logical Address	Transaction Identifier (MS)	Transaction Identifier (LS)	Reserved = 0
Data Length (MS)	Data Length	Data Length (LS)	Header Checksum
Data	Data	Data	Data
Data	Data	Data	Data
Data	Data Checksum	EOP	

Last byte transmitted

Bits in Packet Type / Command / Source Address Extra Length Byte

MSB							LSB
Reserved = 0	Response = 0	Read = 0	Read = 0	Read = 1	Increment/ No inc. target	Extra Source Addr Words	Extra Source Addr Words
Packet Type		Command				Source Address Extra Length	

Figure 6-9 Read Reply Format

The Source Path Address bytes contain any required path address bytes needed to route the reply packet from the destination node back to the source node. The value of the Source Path Address bytes are as specified in the Extra Source Address field of the read command. If logical addressing is being used then the Source Path Address bytes are not present in the reply to the write command. Any Source Path Address bytes are stripped off by the time the reply reaches the source of the write command.

The Source Logical Address byte contains the logical address of the source of the read command packet, as specified in the read command Source Logical Address field.

The Protocol Identifier byte is set to the value 1 (01h) which is the Protocol Identifier for the Remote Memory Access protocol.

The Packet Type field is 00b to indicate that this is a reply packet.

The Command and Source Address Length field are set to the same values as in the command byte of the read command.

The Status byte provides the status of the read command. This is set to zero if the command executed successfully and to a non zero error code if there was an error. See sub-clause 6.6 for a description of the possible error codes.

The Transaction Identifier bytes are set to the same value as provided in the read command. This is so that the source of the read command can associate the reply and data in the reply with the original read command.

The three Data Length bytes contain the length, in bytes, of the data that is to be read and returned in the reply packet. The first byte sent is the most significant byte of the data length. If the read reply packet is indicating an error, i.e. the status byte is non-zero, then the Data Length will normally be zero and there will be no data.

The Header Checksum byte is an 8-bit checksum used to confirm that the header of the reply packet has been received without error.

The Data bytes contain the data that has been read from the memory of the destination node. When reading from memory organised in words (e.g. 32-bit words) then the first byte sent is the most-significant byte of the word.

The Data Checksum byte contains an 8-bit checksum used to confirm that the data was correctly transferred.

EOP character is the End Of Packet market of the SpaceWire packet.

6.4.3 Read action

The operation of the read command is illustrated in the sequence diagram of Figure 6-10.

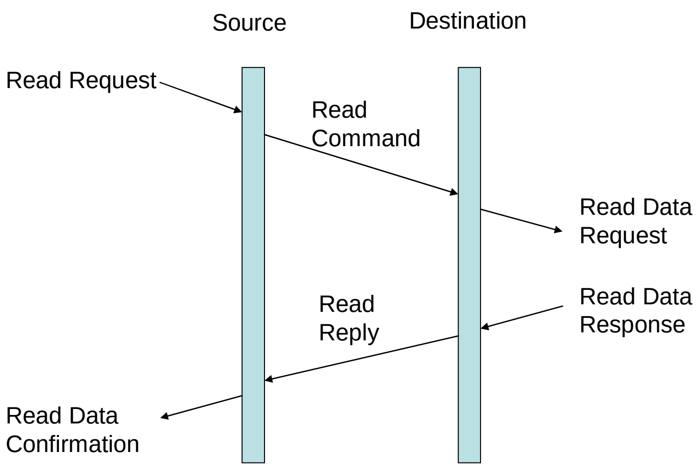


Figure 6-10 Read Command/Reply Sequence

The read command sequence starts when an application requests to perform a read operation (Read Request). The read command is constructed and sent to the destination node (Read Command). When the read command arrives at the destination it is flagged to the user application on the destination node (Read Data Request). The header of the read reply packet is formed and the requested data appended to it. The read reply containing the data is then sent back to the source of the read command. When it arrives there the user application that requested the data is informed (Read Data Confirmation).

6.4.4 Read errors

There are three principal types of error that can occur when executing a read command: read command error, read authorisation rejection, read reply header error and read reply data error. These errors will now be considered.

The sequence of events following a read command error are illustrated in Figure 6-11.

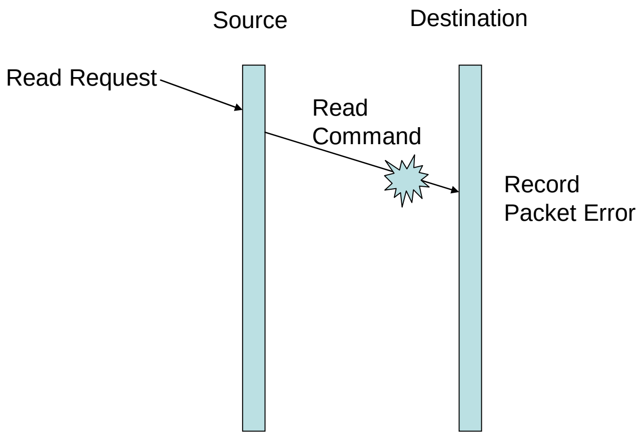


Figure 6-11 Read Command Error

If the read command is corrupted but arrives at the destination node then a packet error will be recorded at the destination, but no other action will be taken by the destination node. It will not read any data and will not return a read reply packet. If the read command is lost altogether then the destination node would know nothing about the read command at all and would not be able to record a packet error.

If indication of this type of error is required at the source node then it is up to the user application at the source to set a timeout timer for the reply to the read command.

A read command may be received correctly (no header checksum error) but may still be rejected by the destination node. For example the read command may be for a different device type than that of the destination node, or the read command may be requesting data from an invalid memory address within the destination node. This situation is illustrated in Figure 6-12.

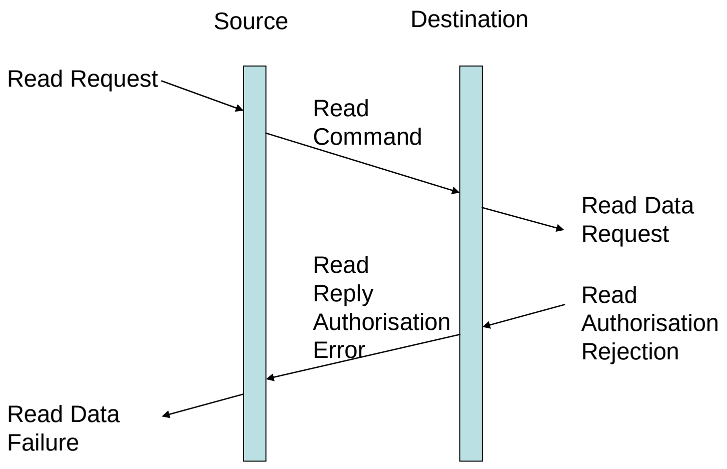


Figure 6-12 Read Authorisation Rejection

When the read command arrives without error at the destination node its parameters are passed to the user application in the destination for authorisation. The read request, in this case, is rejected (Read Authorisation Rejection) and an error message is sent back to the source node (Read Reply Authorisation Error). When this error message arrives at the source node it causes a read data failure to be flagged to the user application in the source node.

The situation that arises following a read reply header error is shown in Figure 6-13.

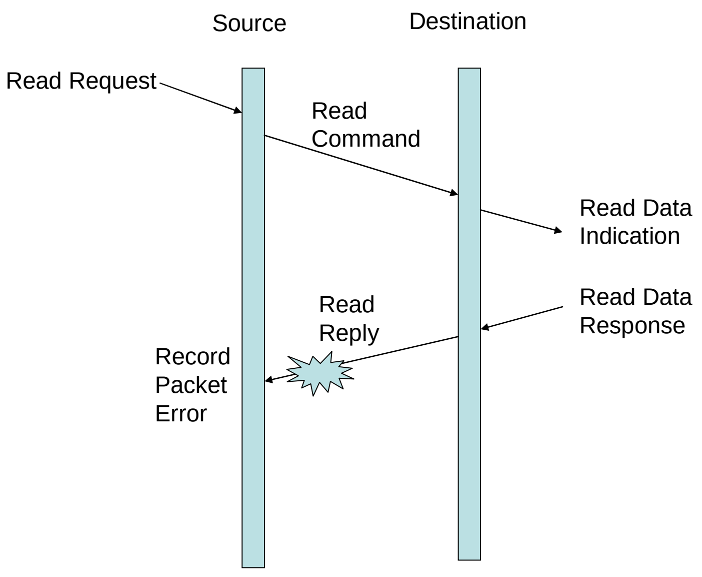


Figure 6-13 Read Reply Header Error

The read command is received by the destination node and a reply returned to the source node containing the requested data. Either the reply packet gets lost altogether or the header of the read reply is received corrupted and a packet error is recorded at the source. Because there is an error in the header it is not known for certain what transaction identifier the reply packet is for, so nothing else can be done by RMAP.

If the user application at source has set a timeout timer for the read reply, then it will be able to detect the missing response, but this is outside the scope of the RMAP.

The result of an error in the data field of a read reply is illustrated in Figure 6-14.

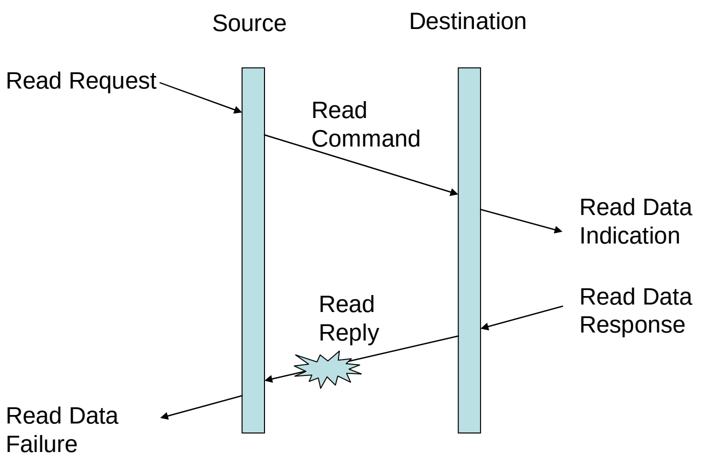


Figure 6-14 Read Reply Data Error

If the header of the read reply packet is received intact but the data field is corrupted as indicated by an incorrect data field length (too long or too short) or by a checksum error, then an error can be flagged to the application immediately (Read Data Failure) without having to wait for a timeout.

6.4.5 Read command parameters

The Read Request has to provide the following parameters:

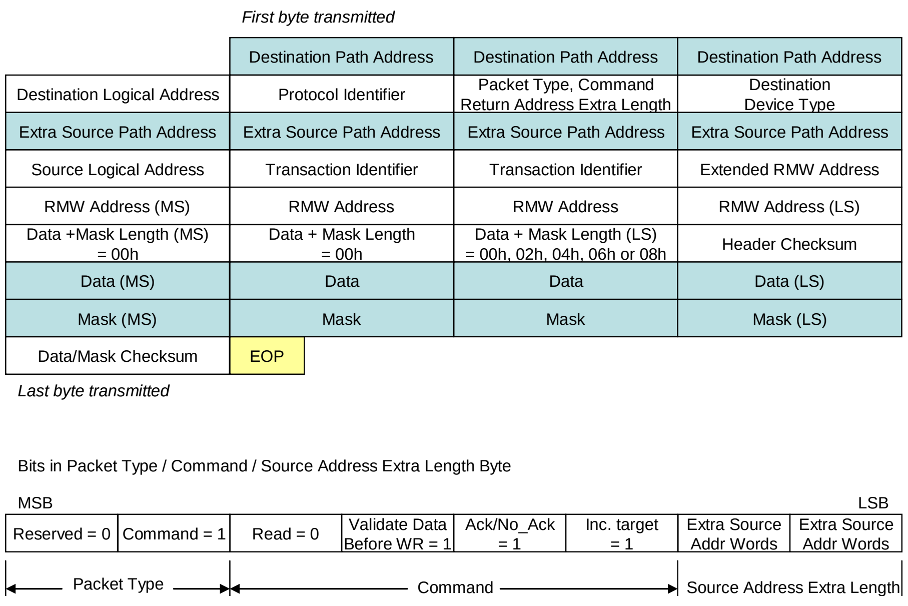
- Destination address
- Source address
- Transaction identifier
- Destination device type
- Read command options
- Read address
- Data length

Note that RMAP does not handle the user application receive buffers, otherwise it would have to maintain at least a pointer for every outstanding read request. It is up to the user application to handle any receive buffers. The appropriate receive buffer for a read reply may be identified in the user application by the transaction identifier in the read reply.

6.5 Read-Modify-Write Command

6.5.1 Read-modify-write command format

The read-modify-write command provides a means for a source node, to read a 32-bit memory location in a destination node, modify some of the bits read and then write the new value back to the same memory location. The original value read from memory is returned to the source node. The format of the command is shown in Figure 6-15.



First byte transmitted

	Destination Path Address	Destination Path Address	Destination Path Address
Destination Logical Address	Protocol Identifier	Packet Type, Command Return Address Extra Length	Destination Device Type
Extra Source Path Address	Extra Source Path Address	Extra Source Path Address	Extra Source Path Address
Source Logical Address	Transaction Identifier	Transaction Identifier	Extended RMW Address
RMW Address (MS)	RMW Address	RMW Address	RMW Address (LS)
Data +Mask Length (MS) = 00h	Data + Mask Length = 00h	Data + Mask Length (LS) = 00h, 02h, 04h, 06h or 08h	Header Checksum
Data (MS)	Data	Data	Data (LS)
Mask (MS)	Mask	Mask	Mask (LS)
Data/Mask Checksum	EOP		

Last byte transmitted

Bits in Packet Type / Command / Source Address Extra Length Byte

MSB							LSB
Reserved = 0	Command = 1	Read = 0	Validate Data Before WR = 1	Ack/No_Ack = 1	Inc. target = 1	Extra Source Addr Words	Extra Source Addr Words
Packet Type		Command				Source Address Extra Length	

Figure 6-15 Read-Modify-Write Command Format

The Destination Path Address and Destination Logical Address are the same as for a read or write command.

The Protocol Identifier byte is set to the value 1 (01h) which is the Protocol Identifier for the Remote Memory Access protocol.

The Packet Type field is 01b, i.e. the command/response bit is set, to indicate that the packet is a command packet, rather than a reply packet.

The Command field holds the read-modify-write command.

The Write/Read bit is zero for a read-modify-write command.

The Validate Data Before Write bit is set (1) so that the data is always validated before it is used to update the memory location. This also distinguishes a read-modify-write from a read command.

The Ack/No_Ack bit is set (1) so that a reply to the read-modify-write command is always produced. This reply will contain the data initially read from the register in the destination node.

The “Increment / No Increment Target” bit is set (1) so that the target memory address is incremented if the width of the memory is less that four bytes (32-bits). This means that when more than one byte is to be read-modified-written the address will be incremented if byte wide memory is being used. Note that the width of the memory word is determined by the target unit and can be any multiple of 8-bits. For example, if the width of the target unit memory word is 32-bits then four data bytes from the data field of the command are read and written into one memory location in the target unit.

The Source Address Length field has the same function as for the read and write commands. If specifies the number of extra 32-bit words needed to hold any source path address.

The Device Type, Extra Source Path Address, Source Logical Address, and Transaction Identifier bytes have the same function as for the read and write commands.

The Extended RMW Memory Address byte holds the most-significant 8-bits of the memory address to be read-modified-written. This effectively extends the 32-bit memory address to 40-bits allowing access to 1 Terabyte of memory space in each node. This byte is set to zero if extended memory addressing is not being used.

The four RMW Memory Address bytes hold the bottom 32-bits of the memory address which is to be read-modified-written. The first byte sent in the command is the most significant byte of the address. When combined with the Extended Memory Address byte a 40-bit memory address is provided.

The three Data Length bytes contain the length of the data that is to be written. In a read-modify write command this gives the total length of data (data and mask) sent in the command, which is twice the amount of data to be read and written. For example if a 2-byte word is to be written, then the data length will be 04h. There will be two data bytes and two mask bytes in the command. Two bytes will be read from memory and returned to the source node. Two bytes will be written combining the read data, the data from the command and the mask. The maximum amount of data that can be read-modified-written with a read-modify-write command is 4 bytes. Hence the data length can only take on values of 00h, 02h, 04h, 06h or 08h. The first byte sent is the most significant byte of the data length. If an invalid data length (01h, 03h, 05h, 07h or >08h) is specified then an error will be returned to the source.

The Header Checksum byte is an 8-bit checksum used to confirm that the header is correct before executing the command.

The Data bytes contain the data that is to be combined with the data read from memory and the mask, and then written into the memory of the destination node. When writing to memory organised in words (e.g. 32-bit words) then the first byte sent is the most-significant byte of the word.

The Mask bytes are use to define how the data to be written to memory is formed. Data to be written is selected on a bit by bit basis from the data send in the command when the corresponding mask bit is set (1) or from the data read in the reply when the mask bit is clear (0).

Written Data = (Mask AND Command_Data) OR (/Mask AND Read_Data).

An example is given in Figure 6-16.

The Data/Mask Checksum byte contains an 8-bit checksum used to confirm that the data and mask information was correctly transferred. The read-modify-write command will only be executed if there is no error in the data/mask.

EOP character is the End Of Packet market of the SpaceWire packet.

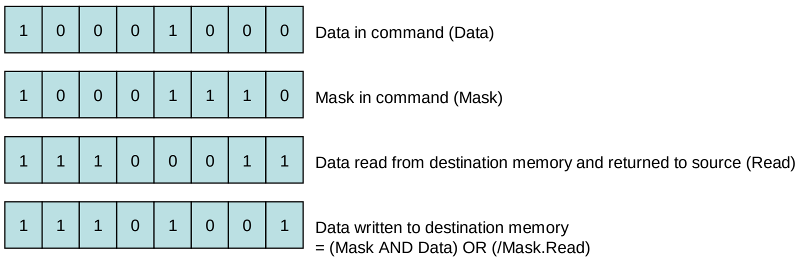
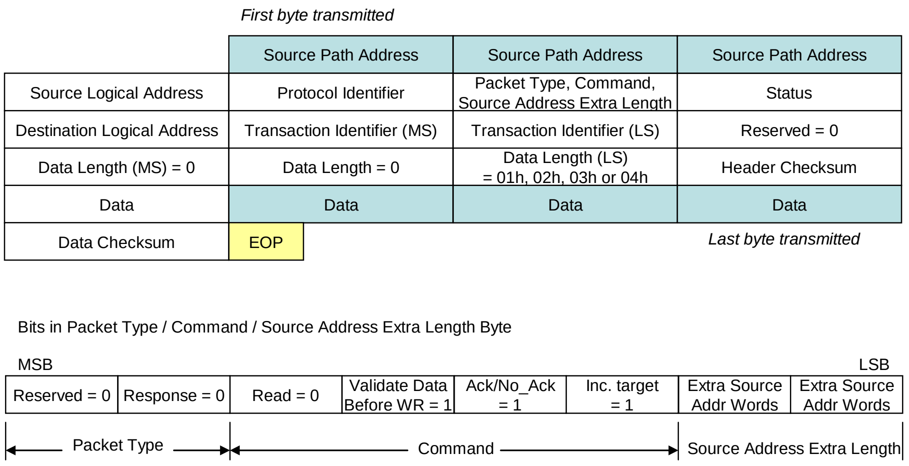

1 0 0 0 1 0 0 0 Data in command (Data)
1 0 0 0 1 1 1 0 Mask in command (Mask)
1 1 1 0 0 0 1 1 Data read from destination memory and returned to source (Read)
1 1 1 0 1 0 0 1 Data written to destination memory
= (Mask AND Data) OR (/Mask.Read)

Figure 6-16 Operation of Read-Modify-Write Command

6.5.2 Read-modify-write reply format

The reply to a read-modify-write command is sent by the destination back to source of the command. The reply is used to indicate the success or failure of the read-modify-write command and to return the data originally read from the destination memory. The format of the write reply is shown in Figure 6-17.



First byte transmitted

	Source Path Address	Source Path Address	Source Path Address
Source Logical Address	Protocol Identifier	Packet Type, Command, Source Address Extra Length	Status
Destination Logical Address	Transaction Identifier (MS)	Transaction Identifier (LS)	Reserved = 0
Data Length (MS) = 0	Data Length = 0	Data Length (LS) = 01h, 02h, 03h or 04h	Header Checksum
Data	Data	Data	Data
Data Checksum	EOP		

Last byte transmitted

Bits in Packet Type / Command / Source Address Extra Length Byte

MSB							LSB
Reserved = 0	Response = 0	Read = 0	Validate Data Before WR = 1	Ack/No_Ack = 1	Inc. target = 1	Extra Source Addr Words	Extra Source Addr Words

Figure 6-17 Read-Modify-Write Reply Format

The Source Path Address bytes contain any required path address bytes needed to route the reply packet from the destination node back to the source node. These bytes are optional and are only present if path (or regional logical) addressing is being used.

The Source Logical Return Address byte contains the logical address of the source of the read-modify-write command packet, as specified in the command Source Address field.

The Protocol Identifier byte is set to the value 1 (01h) which is the Protocol Identifier for the Remote Memory Access protocol.

The Packet Type field is 00b to indicate that this is a reply packet.

The Command and Return Address Length field are set to the same values as in the command byte of the read-modify-write command.

The Status byte provides the status of the read-modify-write command. This is set to zero if the command executed successfully and to a non zero error code if there was an error. See error codes sub-clause 6.6.

The Transaction Identifier bytes are set to the same value as provided in the read-modify-write command. This is so that the source of the command can associate the reply with the original read-modify-write command.

The three Data Length bytes contain the length, in bytes, of the data that is to be read and returned in the reply packet. The first byte sent is the most significant byte of the data length. For a read-modify-write command the data length can be 1,2,3 or 4 only. If the read reply packet is indicating an error, i.e. the status byte is non-zero, then the Data Length will normally be zero and there will be no data.

The Header Checksum byte is an 8-bit checksum used to confirm that the header of the reply packet has been received without error.

The Data bytes contain the data that has been read from the memory of the destination node. When reading from memory organised in words (e.g. 32-bit words) then the first byte sent is the most-significant byte of the word.

The Data Checksum byte contains an 8-bit checksum used to confirm that the data was correctly transferred.

EOP character is the End Of Packet market of the SpaceWire packet.

6.5.3 Read-modify-write action

The operation of the read-modify-write command is illustrated in the sequence diagram of Figure 6-18.

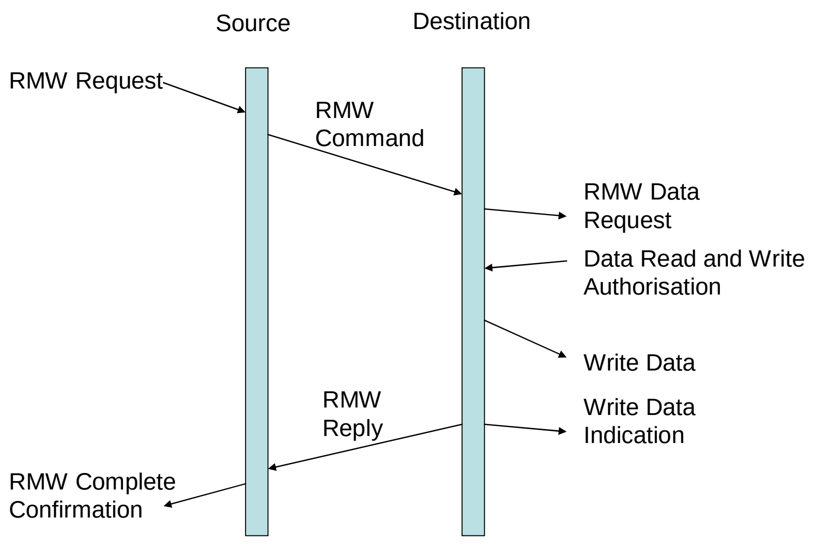


Figure 6-18 Read-Modify-Write Command/Reply Sequence

The read-modify-write command sequence begins when an application requests to perform a read-modify-write operation (RMW Request). In response to this the source node builds the RMW command and sends it across the SpaceWire network to the destination node (RMW Command). When the RMW Command arrives at the destination, the header and data fiels (including the mask bytes) are first checked for errors, since the Validate Before Write bit is always set in the RMW command. If the header and the data do not contain any errors then the user application at the destination node is asked if it will accept the RMW operation (RMW Data Request). If the user application accepts the request it will read the memory location(s) specified in the RMW command and return the data to RMAP (Data Read and Write Authorisation). The data to be written to the memory locations is then calculated from the data read from memory and the data and mask fields of the RMW command. The new data is then written to the memory location(s) that was previously read.

Once data has been written to memory the user application running on the destination node is informed that a RMW operation has taken place (RMW Indication). Since the acknowledgement bit (Ack/No_Ack) is always set for a RMW command, a reply will be sent back to the source of the command containing the data originally read from the destination memory (RMW Reply). When the write reply is received, the source node indicates successful completion of the write request (RMW Complete Confirmation).

6.5.4 Read-modify-write errors

There are four principal types of error that can arise during a read-modify-write operation: RMW Command Error, RMW Authorisation Rejection, RMW Reply Header Error and RMW Reply Data Error.

The sequence of events that occurs when there is an error in the header of the RMW command is illustrated in Figure 6-19.

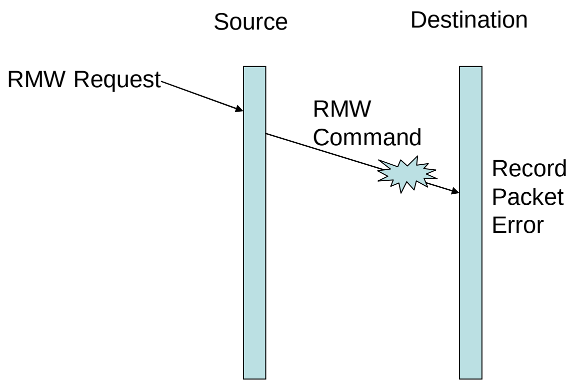


Figure 6-19 Read-Modify-Write Command Header Error

The RMW command packet arrives at the destination and its header is found to be in error. This fact is added to the error statistics in the destination node and the packet is discarded. No other action is taken at the destination or source nodes.

The situation that arises when there is an error in the data field of the read-modify-write command is shown in Figure 6-20.

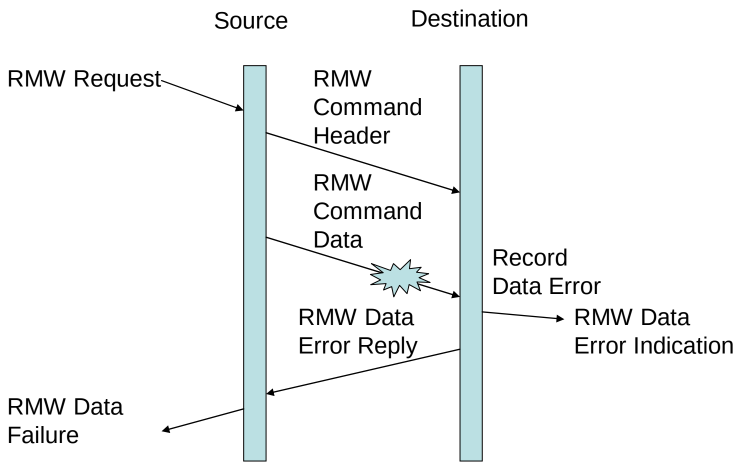


Figure 6-20 Read-Modify-Write Command Data Error

The header of the RMW command has been received without error but the data checksum indicates that there has been an error in the data field. A data error is recorded in the destination node. The user application in the destination node is informed that a RMW command has been received with corrupted data. Since the header of the RMW command was intact it is also possible to report the error back to the source. A RMW reply packet containing the appropriate error code is sent back to the source node (RMW Data Error Reply). When this is received at the source node the error is reported to the user application (RMW Data Failure). RMAP returns the error code and the transaction identifier to the source node so that the user application can determine the original of the RMW command and the type of error that occurred.

If the RMW command is valid, the user application at the destination node is asked if it will accept the RMW operation (RMW Data Request). If it rejects the RMW operation (RMW Authorisation Rejection) then an RMW error reply is returned to the source node (RMW Reply Error). This situation is illustrated in Figure 6-21. When the RMW Reply containing the error code is received back at the source node, a RMW error indication (RMW Failure) is signalled to the user application in the source node.

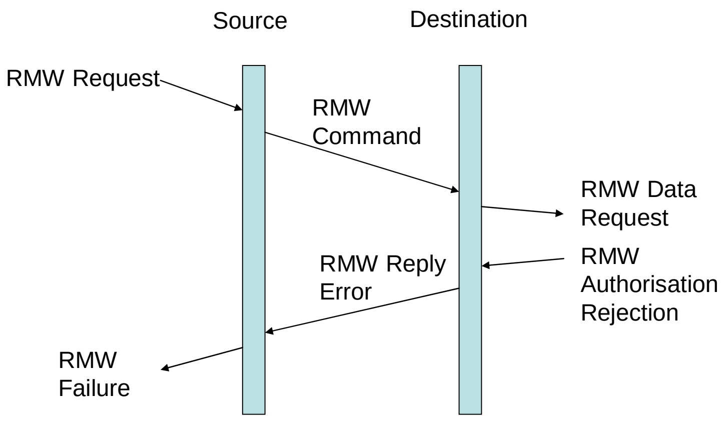


Figure 6-21 Read-Modify-Write Authorisation Rejection

It is possible that the write reply is corrupted or for some other reason does not reach the source node intact. This situation is illustrated in Figure 6-22.

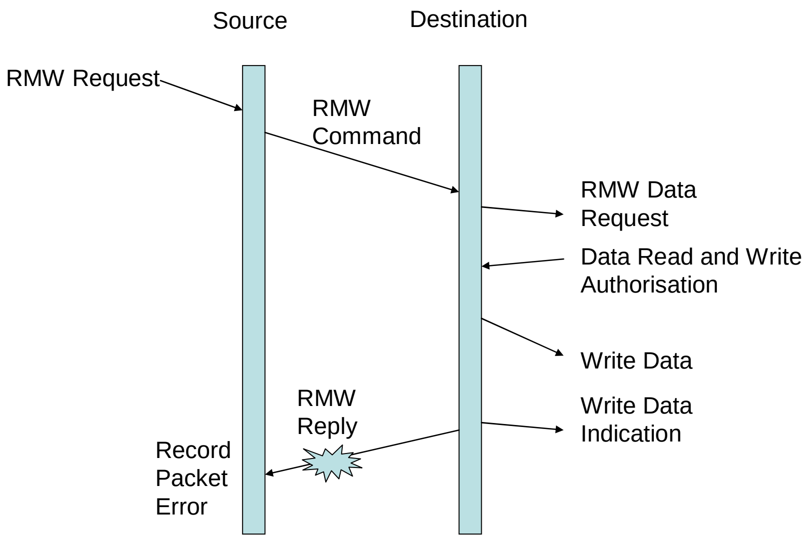


Figure 6-22 Read-Modify-Write Reply Error

The data has been correctly written into destination memory and the destination application has been informed. The RMW reply that is sent back to the source node is corrupted. If the corrupted packet arrives at the source node (or indeed any other node) it is recorded as a packet receive error.

6.5.5 Read-modify-write request parameters

The Read-Modify-Write Request has to provide the following parameters:

- Destination address
- Source address
- Transaction identifier
- Destination device type
- RMW command
- Memory address
- Data length
- Data
- Mask

6.6 Error codes

The possible error codes that can arise are listed in Table 6-1. These error codes are returned in the status field of any reply including acknowledgements and error replies.

Table 6-1 Error Codes

Error Code	Error	Error Description
000	Command executed successfully	
001	General error code	The detected error does not fit into the other error cases or the node does not support further distinction between the errors
002	RMAP command not supported by node	The header checksum was decoded correctly but the command byte is not accepted by the node
003	RMAP device type not supported by node	The header checksum was decoded correctly but the device type does not match that of the destination node.
004	Invalid data checksum	Error in the checksum of the data field
005	Early EOP	EOP marker detected before the end of the data.
006	Late EOP	EOP marker detected beyond the expected end of the data.
007	Verify buffer overrun	The verify before write bit of the command was set so that the data field was buffered in order verify the data checksum before transferring the data to destination memory. The data field was longer than could fit inside the verify buffer resulting in a buffer overrun.
008	Authorisation failure	The destination user application did not authorise the requested operation
009	RMW data length error	The data in a RMW command does not match the data length field or is invalid (01h, 03h,

		05h, 07h or >08h(.

6.7 Device types

Currently six device types have been defined:

- 00h is a general device type
- 01h is a router
- 02h is a sensor
- 03h is a mass memory unit
- 04h is a processing element
- 05h is network controller

Note that a node may be able to act as more than one device type. For example a processing node may be able to act as a processing element and a mass memory unit. In this case it would be able to respond to more than one device type.

The general device type is used when a SpaceWire device does not fit into any of the other defined device categories.